TAO - Toolkit for Advanced Optimization
Tutorial and Exercises

Workshop on the ACTS Toolkit
August 18–21, 2008
National Energy Research Scientific Computing Center

1. Locate the TAO and PETSc documentation at

   ```
   http://www.mcs.anl.gov/tao
   http://www.mcs.anl.gov/petsc
   ```

2. On bassi, set the environmental variables `TAO_DIR`, `PETSC_DIR`, `PETSC_ARCH` using the commands:

   ```
   > module clear
   > module load tao/1.10
   ```

3. Create a subdirectory such as `taoexamples`, enter it, and copy several example problems to the new directory using the commands:

   ```
   > mkdir taoexamples
   > cd taoexamples
   > cp -R $TAO_DIR/src/unconstrained/examples/tutorials/* .
   > ls
   ```

   There should be a `makefile` and several examples ending in `.c` and `.F`.

4. Run an example with TAO. We are going to use TAO to minimize the function

   $$f(x_1, x_2) = 99 * (x_2 - {x_1}^2)^2 + (1 - x_1)^2$$

   on a single processor.

   - Compile the program using
     `> make rosenbrock1` (or `make rosenbrock1f` for Fortran users)
   - Execute the program with
     `> mpiexec -n 1 ./rosenbrock1 -tao_monitor -tao_view`
     (or `mpiexec -n 1 ./rosenbrock1f -tao_monitor -tao_view`)

     What method was used to solve the problem? What is the function value at the final iterate? How many iterates were used to reach the solution? How many function evaluations?

5. Another TAO example finds the minimum surface area of an object over a two-dimensional domain in accordance with some boundary conditions.

   - Compile and execute the `minsurf2` example using

     ```
     > make minsurf2
     > mpirun -np 2 ./minsurf2 -tao_monitor -tao_view
     ```

   - This problem uses the variables `mx` and `my` to determine the discretization of the grid. By default, these values are set to 4 ($4 \times 4 = 16$ variables). Increase the discretization of the domain by using the command

     ```
     > mpirun -np 2 ./minsurf2 -tao_monitor -tao_view -mx 20 -my 20
     ```

     How does this affect the solution? Try different solvers using the option `-tao_method` followed by `tao_cg`, `tao_lmvm`, `tao_ntr`, or `tao_nls`. How many iterations do the different solvers require to solve the problem?

   - Set the tolerances to increase the precision of the solution by adding the command line options `-tao_frtol 0 -tao_fatol 1e-8`

     Now how many iterations does each solver take?

   - Execute the programs from the last step again, but this time use the command line option `-log_summary` to get detailed performance information.

     Look under the PETSc Performance Summary section and determine how long each algorithm takes to solve the problem. How many floating point operations (flops) are required?

   - Run the problem `minsurf2.c` on four processors and view the output.

     ```
     mpirun -np 4 ./minsurf2 -tao_monitor -mx 20 -my 20 -log_summary
     ```

6. Having the correct gradient and Hessian is critical for the success of a TAO application. In this section we will explore the built-in tool for validating the gradient subroutines.

   - Open the file `minsurf2.c` and introduce a small error to the FormFunctionGradient() subroutine (for example, on line 314 change `df1dxc = d1*hydhx` to `df1dxc = 0.8*d1*hydhx`).

   - Compile the altered file (`make minsurf2`) and run again (`mpiexec -n 2 ./minsurf2 -tao_monitor -tao_view`). Depending on the change you made, TAO may or may not converge to a solution. You can check the correctness of the new function/gradient routine by using the TAO utility solver `tao_fd_test`:

     ```
     > mpiexec -n 2 ./minsurf2 -tao_method tao_fd_test
     ```

     TAO has compared the programmed gradient routine against a gradient computed using finite differences at the programmed starting point and two other points and displayed the results. According to the message accompanying the results, the difference between the hand-coded gradient and the finite-differences gradient should be very close to zero.

   - Try fixing the introduced error, compile and rerun, and confirm that for the correct code the difference is close to zero. You can can also have TAO check the correctness of the Hessian code by adding the option `-tao_test_hessian`.